[redacted]

Math

UIUCDCS-R-72-507

COO-1469-0201

ITEM ANALYSIS

by

JOHN ALLEN KOCH

March 1972

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN · URBANA, ILLINOIS

UIUCDCS-R-72-507

ITEM ANALYSIS*

by

JOHN ALLEN KOCH

March 1972

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS
URBANA, ILLINOIS 16801

* Supported in part by the Department of Computer Science and the Atomic Energy Commission under grant US AEC AT(11-1)1469, and submitted in partial fulfillment of the requirements of the Graduate College for the degree of Master of Science in Computer Science.

ACKNOWLEDGMENT

I would like to thank the following people whose assistance is greatly appreciated. To my advisor, Professor C. W. Gear, for his aid and suggestions. To Mr. Al Whaley for a starting point and his help with innumerable ABENDs. To Miss Barbara Hurdle for her typing of this thesis and for her excellent figures. And, finally, to the Department of Computer Science and the Atomic Energy Commission for supporting this thesis.

PREFACE

This thesis describes Item Analysis, a program that analyzes a user's description of a network and converts it into a structure that is suitable for analysis. This program is contained in the graphics-oriented simulation and modeling system developed at the University of Illinois. This report also contains documentation of the program which may not be of interest to the reader who only needs a broad description of the function of Item Analysis. Such readers are encouraged to cover only the first section under each major heading.

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

Item Analysis is a part of the general simulation and modeling system developed at the University of Illinois. The total system allows a user, through interactive graphics, to define a model for a proposed or existing system. The model consists of a network of elements and its corresponding equations. Each element has terminals which can pass variables (e.g., voltage, current, pressure) to other elements. In a network, terminals are connected together at nodes. In a recursive manner, elements can be networks of other elements. For example, current sources and diodes could be defined as elements. The equivalent circuit for a transistor could be constructed as a network of current sources and diodes. Then this "transistor" could be used in another network (which might also contain diodes and current sources). User element and network definitions are kept in a file and can be used or modified at any time. It is possible for a network to be solved for "steady state conditions (all derivatives zero), dynamic behavior (numerical integration), or small perturbation response (eigenvalues and eigenvectors of the Jacobian of the differential equations)" (see [1], [2]).

When the user is ready to have a network analyzed, a call is made to Item Analysis. The data structures that have been created in the PDP-8 to represent the network are sent to the IBM 360 and through the GRASS supervisory system (see [3]). Item Analysis works upwards from the basic (not decomposable) elements to the entire network transforming the structure into one suitable for use by Global Analysis (see Figure 1) which works in the opposite direction. Variable name tables

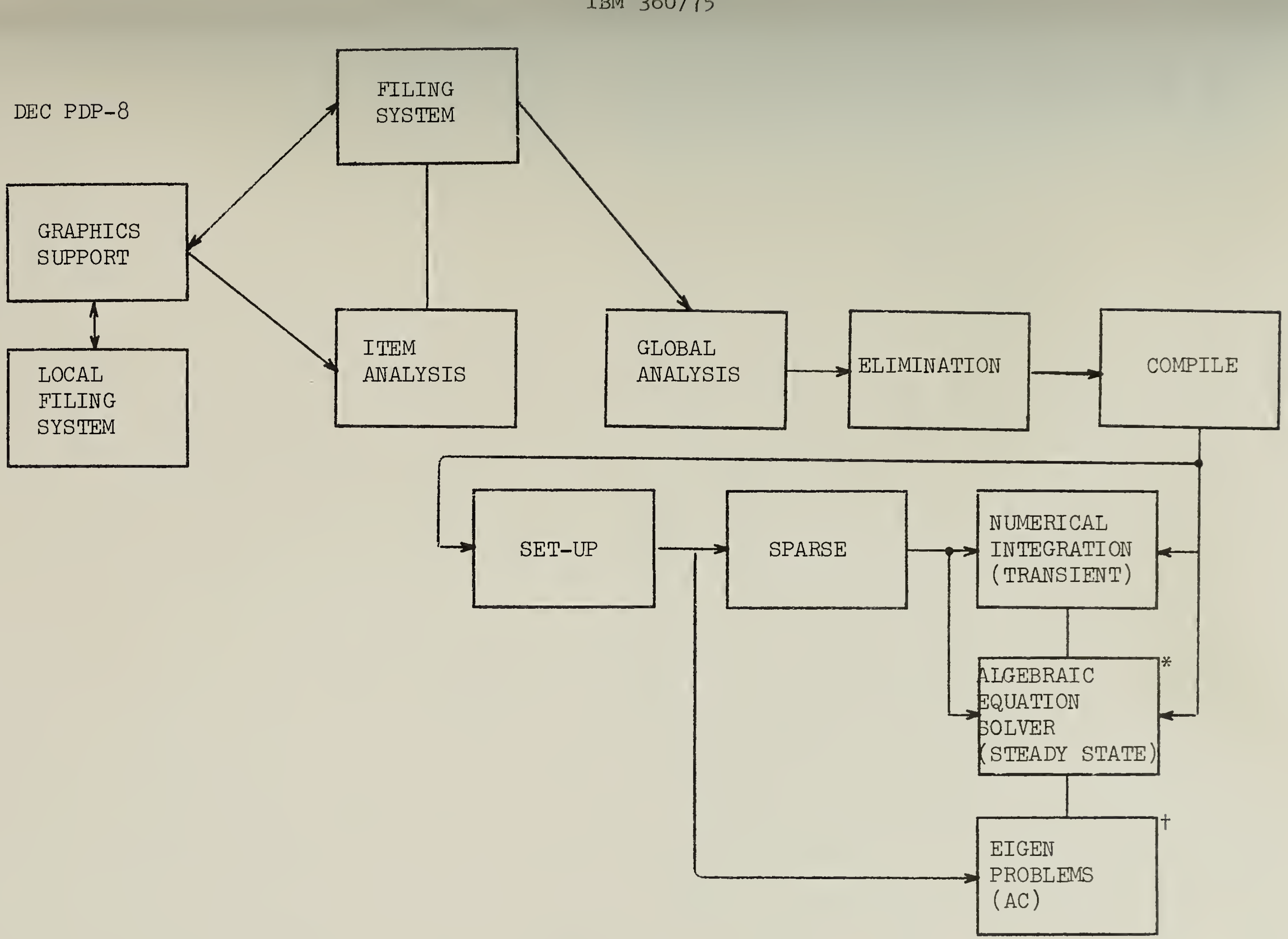


Figure 1. Simulation and Modeling System

are created for each element which is referred to in the equation trees. All of the equations are parsed and placed in coded form in binary trees. Item Analysis detects syntax errors in equations and gross errors in element definition and informs the user. Tables are created in the output which can later be used in communicating results or errors detected in equations that have been modified at almost every level in the system. Each element in a network must be put through Item Analysis before the complete network can be analyzed. Once they have been, however, the elements need never be put through Item Analysis again if their definition remains unchanged. Thus, a resistor can be put through Item Analysis once and then used in as many networks as desired without "re-Itemizing" the resistor for every network. When each element and the network has been through Item Analysis, control is then passed to Global Analysis which leads to the remainder of the simulation system.

2. INPUT/OUTPUT COMMUNICATIONS

2.1 Remote Monitor

Item Analysis operates in the 360 under a two level monitor (see Figure 2). The top level, G8OPERAT, handles the actual data transmission over the 2701 data link between the PDP-8 and the 360 and can attach or detach the second level upon command. Thus, G8OPERAT is the only program that must be running when a user attempts to communicate with the 360. The second level monitor, SPACT, handles the attaching or detaching of user program modules (e.g., Item Analysis). The S8 macros (section 2.2) are used to communicate between the user and Item Analysis through SPACT and G8OPERAT. Commands and input structures are sent from the PDP-8 and error messages are sent from Item Analysis to the user in this manner. Since SPACT and G8OPERAT do not examine or modify the records they transmit, Item Analysis must ensure that each record that is sent or received is in the proper format.

Output data structures are handled by the filing system through use of the XFILE macros (section 2.3). Four files are kept for each user. When a picture is sent from the PDP-8, it is immediately stored in PICLIB. This allows the user to re-analyze a picture at a later date without having to re-transmit it from the PDP-8; it is simply retrieved from the file. After Item Analysis is complete, the output data structure is placed in ITEMLIB. There the network can be accessed as input for Global Analysis and can be used to translate internal code names into ones that the user can identify. Declarations of node type are written out to NODLIB. The last file is ERRLIB which contains all the syntax errors detected by Item Analysis.

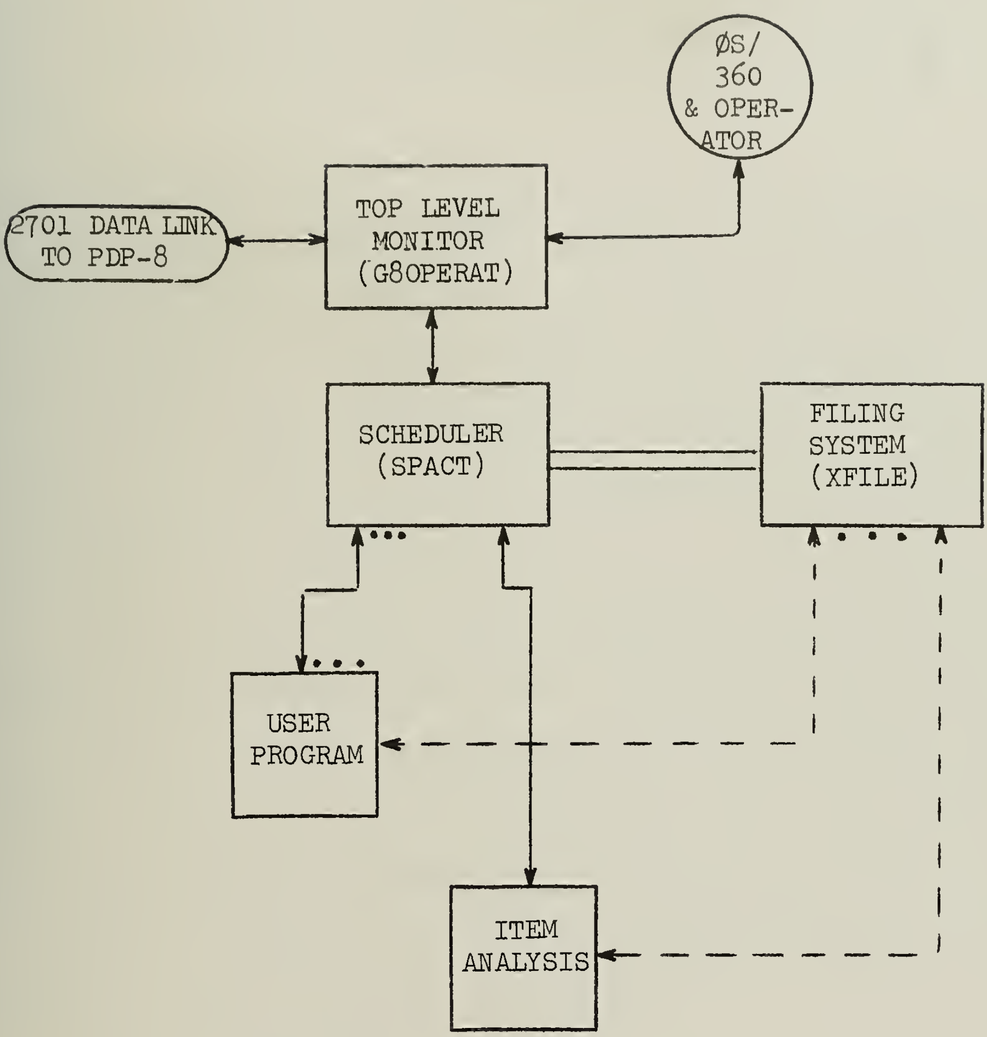


Figure 2. Monitor Linkages

2.2 Input and User Communication

The S8 macros described in detail in [4] are used for input communication. A user logs on to G8OPERAT, starts SPACT and requests that ITEM (Item Analysis) be started. At this point ITEM executes an S8ENTRY macro:

```
S8ENTRY N=(2)
```

This notifies SPACT that ITEM has begun executing and links it to SPACT and XFILE (see Figure 2). 'N=(2)' means that the work area address is in register 2. The macro S8DATA is used to set up this work area in the correct format. Then an S8READ is executed which causes ITEM to wait until a command or a picture has been sent from the PDP-8.

```
S8READ R=(1), HALT=HALT
```

'R=(1)' indicates that register 1 points to the work area and will point to the record read upon return. 'HALT=HALT' means that the location 'HALT' is branched to if the user or the system requests a halt. This S8READ is executed every time ITEM requires action by the user. Note that upon completion of a command or analysis of a picture, the input area (which is GETMAIN-ed by SPACT) is FREEMAIN-ed to keep these used areas from tying up available core memory.

```
S8WRITE BUF,TYPE=0,R=(1),HALT=HALT
```

An S8WRITE is executed when a message is to be sent to the user's terminal. ITEM waits until the message is transmitted. BUFF is the location of the record to be written, TYPE=0 means that 0 is inserted into the type field of the record and R and HALT are as described above. The following is a list of command messages to the user, error messages and their meaning:

Command List

(ALPHA is a valid six character element or network name)

1. 'ITEMIZE ALPHA'--causes ALPHA to be read from PICLIB and put through Item Analysis.
2. 'ANALYZE ALPHA'--causes Item Analysis to call global analysis with the parameter ALPHA. Item analysis does no processing of ALPHA.
3. 'DESTROY ERRLIB'--causes Item Analysis to delete the user's error library. This command is only valid for destroying the error library and will not be recognized if any other file name but ERRLIB is used.
4. 'RESTART ALPHA'--causes Item Analysis to call COG with the parameter ALPHA. The system will restart processing at that point.
5. 'TEST'--allow diagnostic blocks to be dumped on the printer.
6. 'NOTEST'--no dumps are taken. This is the default value.
7. 'LOOP'--set bit which causes Global Analysis to generate loop equations.
8. 'NOLOOP'--causes the bit to be turned off so no loop equations are generated. This is the default value.

Messages to User

1. 'SYNTAX ERRORS DETECTED.'

   If errors detected by Item Analysis.

2. 'UNKNOWN COMMAND.'

   An illegal command was received.

3. 'INVALID BLOCK TYPE RECEIVED.'

   An error in transmission occurred.

4. 'ITEM READY FOR FIRST INPUT'

   Indicates Item Analysis is loaded and ready.

5. 'ITEM ANALYSIS COMPLETE'

   The analysis of an element is done and Item Analysis is ready for another input.

6. 'RECORD XXXXXX-XXXX NOT FOUND. ENTER AGAIN.'

   When an 'ITEMIZE' command is given and the indicated block does not exist on PICLIB.

7. 'MNEMONIC STORED.'

   After user has sent a mnemonic.

8. 'ERRLIB GONE...'

   After the 'DESTROY ERRLIB' command is completed.

Error Messages

1. 'NO : IN NODE TYPE DECLARATION.NAME : E(X,Y), I(P,T9)'

   i.e., 'ELECTRIC E(A,B), I(C,D)'

2. 'NO) IN NODE TYPE DECLARATION.NAME : E(X), I(A,R)'

   i.e., 'ELECTRIC : E(A,B), I(X'

3. 'VARIABLES IN NODE TYPE DECLARATION NOT E OR I.'

   i.e. 'ELECTRIC : W(A,B,C), Z(Q,R,T)'

4. 'VARIABLE REDEFINED.'

   Variable defined twice as either a global, local, or parameter.

5. 'INVALID ENDING CHAR IN PARAMETER DEFINITION.'

   Must end with a single variable or an equation.

6. 'NO E OR I VARIABLES SPECIFIED FOR TYPE.'

   Just a type with no variables was entered. (i.e., 'ELECTRIC :')

7. 'NON-EXISTANT TERMINAL TYPE REFERENCED.'

   When a terminal has been assigned a type but the type was later deleted from the list of types.

8. '******* ILLEGAL XXXXXX DETECTED'

   Message originates in PARSE and XXXXXX refers to the level of compilation at which the error was detected in an equation.

All error messages are also placed in ERRLIB using the filing system.

2.3 Output Communication

The output from Item Analysis that remains in the 360 is placed in the filing system using the XFILE macros[4]. Each file name is 16 bytes long and the convention used is the first eight bytes of the user's log-on name and the second eight bytes are the file name. For example, all of the pictures that JONES sends to ITEM would be stored in the file name.

```
|J|O|N|E|S|b|b|b|P|I|C|L|I|B|b|b|
+0              +8              +16
```

Note that both names are padded on the right with blanks. Records within a file are named using the first record format is as follows:

```
|RECORD NAME | DATA LENGTH (=n) | DATA |
+0           +8                  +10    +(10+n)
```

The determination of record names is different for each of the four files Item Analysis uses. In PICLIB, the six character picture name and a two byte block identifier are used (section 3). NODLIB uses the type name (section 4.3) and ITEMLIB uses the six character picture name with two EBCDIC blanks. In ERRLIB, the record names are supplied by the filing system so that the last error message appears at the end of the file.

Item Analysis uses eleven types of XFILE macros. To use the system, a location called 'ABEND' is provided which causes ITEM to terminate abnormally. This location is branched to by the XFILE macros when an error is detected by the filing system (e.g., 'record not found', 'file not open on attempted read').

```
<name> XLIST NAME=XXXXXXXX<name>,
             NREC=1
```

The XLIST macro creates an area where parameters and status and control codes for each file are stored. There is an XLIST for each of the four files used by ITEM. When doing a filing system operation, values are stored in the XLIST and then the call to XFILE is made. Therefore, only the parameters that were changed from the last call need be specified. For example, since the number of records read or written at a time is always one in ITEM, the value NREC=1 only appears on the XLIST macro and not with each separate read or write.

```
XINIT ITEMLIB
```

This macro causes the filing system to be initialized. This allows the user to access the file defined by the control card:

```
//DISK  DD  DSN=xxxxxxxx,DISP=OLD
```

The ITEMLIB XLIST is used as a scratch area. When control is returned to ITEM, R1 points to the XDSCB, an internal control block. All four parameter lists that are going to be used must be linked together using the folloiwng macro:

```
XLINK <name1>,  {XDSCB=(R1)
                 LIST2= name2}
```

The XDSCB parameter is supplied in the first XLINK since the pointer is in R1 upon return from the XINIT. The LIST2 parameter is then used in the three other XLINK's to link the parameter areas together.

```
XOPEN <name>
```

All files that are to be read from or written to must first be opened. The above macro is used to open each file before any I/O is performed on them.

```
XDELETE PICLIB,BUF=(1),TEST=(¬,RNF)
```

Upon receiving a picture (through an S8READ), Item Analysis does an XDELETE on PICLIB to delete that record if it has previously been sent. BUF=(1) means that the record address is in register 1 (the first eight bytes to the record name). TEST=(¬,RNF) ensures that the system will not abend if the record not found condition occurs. If this macro was not executed, a subsequent attempt to write the same record onto PICLIB would result in an abnormal termination.

```
XREAD PICLIB,BUF=(10),BUFLEN=(7),TEST=(NOGO,RNF)
```

This causes a picture previously sent to the 360 to be read from PICLIB. The parameters BUF=(10) and BUFLEN=(7) indicate the registers where the address of the buffer and its length are to be found. TEST=(NOGO,RNF) causes ITEM to branch to the location 'NOGO' when the record not found condition occurs. Thus, the user has requested a picture that has not been stored on PICLIB (#6 in 'Messages to User').

```
XWRITE <name>,BUF=(10),BUFLEN=(7)
```

When any records are to be written on PICLIB, NODLIB or ITEMLIB, the above macro is used. BUF=(10) and BUFLEN=(7) indicate the registers where the address of the record to be written and the buffer length are located. (Note that BUFLEN is 10 more than the value found at BUF+8).

```
XAPPEND ERRLIB,BUF=(R6),BUFLEN=(R2)
```

The syntax errors detected are written onto ERRLIB with the above macro. The name is supplied by the system but eight bytes for the name must still appear at the beginning of the message. The parameters are the same as in an XWRITE.

```
XCLOSE <name>,TEST=(¬,SEQ)
```

All the files are closed before ITEM is exited. TEST=(¬,SEQ) means that the system will not abnormal end if the file appears to be closed already.

```
XDESTROY ERRLIB
```

This macro is used when the user has examined his error messages and wishes to get rid of the entire error message file (#3 in 'Command List'). This macro is executed with ERRLIB properly linked but not opened.

XTERM ITEMLIB

This is the last call issued to the filing system and does all of the required processing to terminate use of the filing system. ITEMLIB is once again used as a scratch area.

3. INITIALIZATION AND INPUT SPECIFICATIONS

Item Analysis must be initialized when it is first called because of the use of the S8 and XFILE macros for Input/Output. Work areas for these macros are set up and linked together. The user's log-on name is moved into the four XLISTs from the SPACT parameter list (Figure 3). These will be used as file keys (see section 2.3). ITEM is now ready to accept commands, pictures or mnemonics from the user. Note that ITEM is waiting for a user response after issuing an S8READ whenever it is not processing.

```
SPACT

Parmlist for subtask                 Subtask

 +0 (available)                      (user must:
 +4 A(user work area)                1.  save parmlist address
 +8 A(XFILE001)                      2.  set XFILE001 into V-con
+12 A(log user)                      3.  call S8ENTRY
+16 A(parm string)                   before anything else)


Parmstring                           User Work Area (14 words)

24 characters:                        +0 0              SPACT ECB for
   parm field                                           HALT request
   (from START command)               +4 A(2701ØPRQ)    2701 request link
                                      +8 0              2701 Read done ECB
                                     +12 0              2701 Write done ECB
Loguser                              +16 (+8,+12,+0)    S8WAIT ECBLIST
                                     +28 (+8,+0)        S8READ ECBLIST
8 characters:                        +36 (+12,+0)       S8WRITE ECBLIST
  user logon name                    +44 0              A(OTHER)
                                     +48 0              Register 2 save wrd
                                     +52 0              Addressability
Trmcntrl                                                save word

+0 2701ØPRQ: READ request ECB
+4           WRITE request ECB
+8 A(user work area) +8 ØPdone ECB's
```

Figure 3. SPACT/User-Program Linkage After S8ENTRY

When data is sent to the 360 from the PDP-8, the six bit PDP-8 byte is right justified in the eight bit 360 byte. The first two bits are zero. For characters, a simple translate from ASCII to EBCDIC is all that is needed; but for numbers, the bits must be shifted to eliminate the extra zero bits that were inserted. The macro LDP is used in ITEM to shift two bytes around and end up with the correct number.

Bytes 6 and 7 of data sent to the 360 are checked for the type of block. If byte 6=0, the block is a command from the user. If non-zero, then it's a mnemonic if byte 7 is also non-zero. Otherwise, it is the header block for a picture. Each picture is composed of a header and 10 other blocks. Each block has a code in bytes 6 and 7 which is used as the last two bytes of the record key when it is written onto PICLIB (e.g., the header for GRVLNK would be in PICLIB under the record name 'G,R,V,L,N,K,8,0'). The codes for each block are as follows:

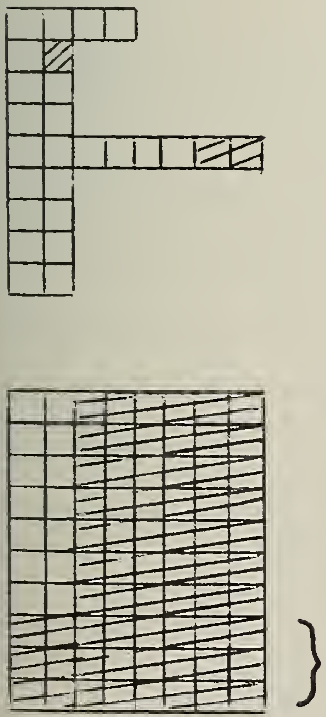
Code (2 bytes)	Block Type
0800	Header
0801	Local lines
0802	Comment text
0803	Subpictures
0804	Spare block
0805	Terminal connections
0806	Declarations
0807	Equations and parameters
0808	Spare blocks
0809	
080A	

The macro PDP8DATA contains DSECTs which describe the relative position of individual fields within the input blocks.

3.1 Header Block

The header block (Figure 4) contains bits that indicate if a block exists. If it does not exist, no attempt is made by ITEM to read it. Even though ITEM does not use the local lines or comment text blocks, these are placed on PICLIB so the user may retrieve a picture from the 360 and have all the blocks necessary to construct it on the screen. Thus, the header block is used to allocate space for reading the rest of the blocks.

▨ ≡ unused byte



length of GETMAINed area
Console number
Header code ('0800')
Length of the following (in words):
Name of picture
Total length of data (in blocks)
PDP-8 address
Control check word
User identification

The following halfwords are (length of blocks)/256. If bit 1=1, no block exists.

Local line block length
Comment text block length
Subpicture block length
Spare
Node connection block length
Declaration block length
Equation block length

Spares

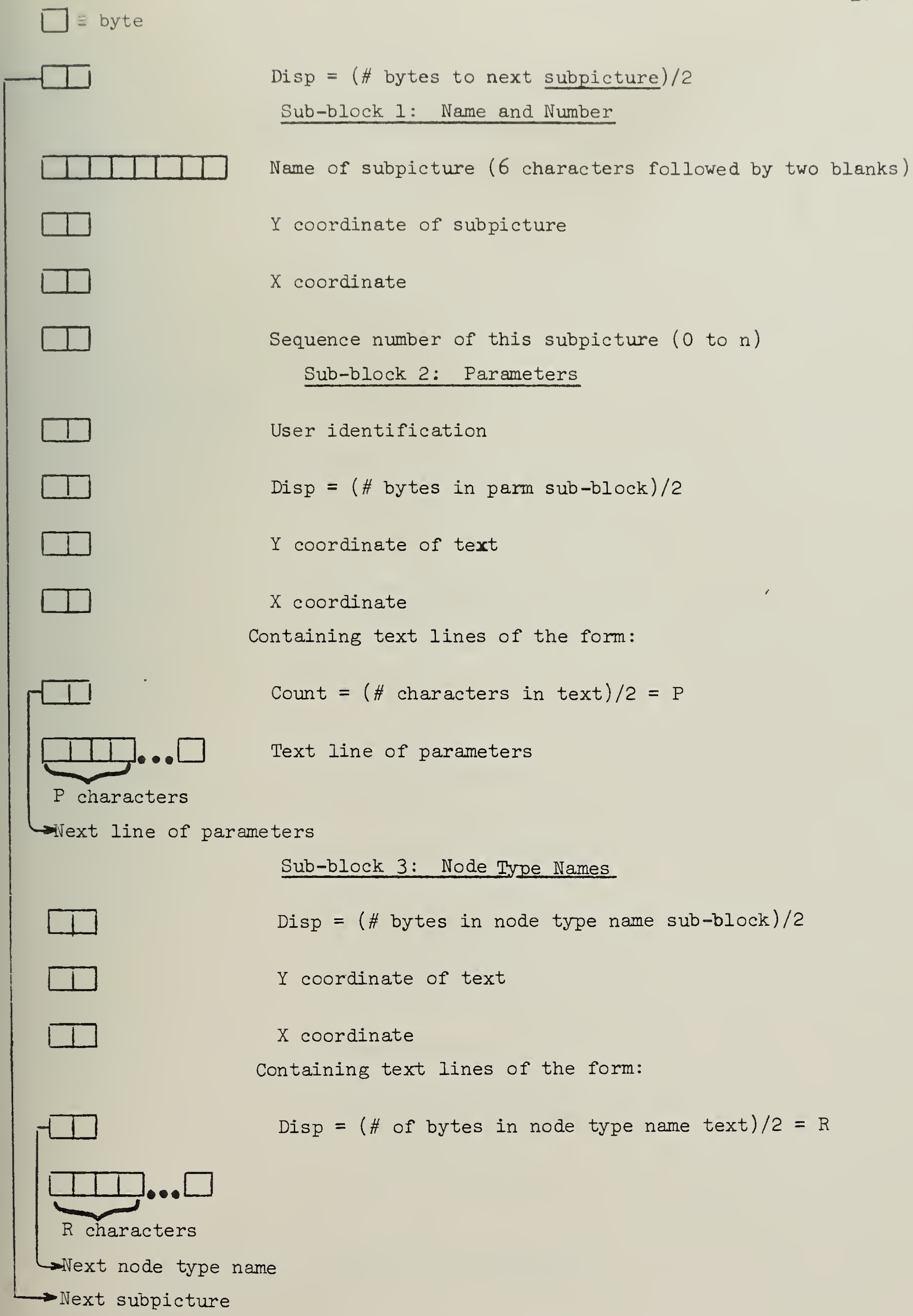
Figure 4. Header Block Layout

3.2 Subpicture Block

The subpicture block (Figure 5) contains three sub-blocks for every instance (occurrence) of an element in a network. The first sub-block contains the name of the element and its sequence number (e.g., the first element put into the network is #0, the second is #1, etc.). ITEM uses this block to create the Element Instance table (section 4.7). The second sub-block contains the parameter assignments for an instance. The third sub-block holds a list of the types of terminals on that subpicture. If there are two or more terminals of the same type on an element, the type name appears only once in this list and the terminals reference its position in the list (e.g., if there are two CONTROL terminals, the name 'CONTROL' appears once in the list, say in position 3--then those terminals reference position 3).

3.3 Terminal Connection Block

The terminal connection block (Figure 6) contains a description of all the terminals in a picture and their interconnections. Entry 1 has the names of all the types of local (not assigned to an instance) terminals. Entry 2 has the connections from terminal NX to terminal MX. This entry is used by ITEM to construct the Node/Connection Table (section 4.7). Finally, every terminal in the picture is described in the remaining entries.



□ ≡ byte

Disp = (# bytes to next subpicture)/2

Sub-block 1: Name and Number

Name of subpicture (6 characters followed by two blanks)

Y coordinate of subpicture

X coordinate

Sequence number of this subpicture (0 to n)

Sub-block 2: Parameters

User identification

Disp = (# bytes in parm sub-block)/2

Y coordinate of text

X coordinate

Containing text lines of the form:

Count = (# characters in text)/2 = P

Text line of parameters

P characters

Next line of parameters

Sub-block 3: Node Type Names

Disp = (# bytes in node type name sub-block)/2

Y coordinate of text

X coordinate

Containing text lines of the form:

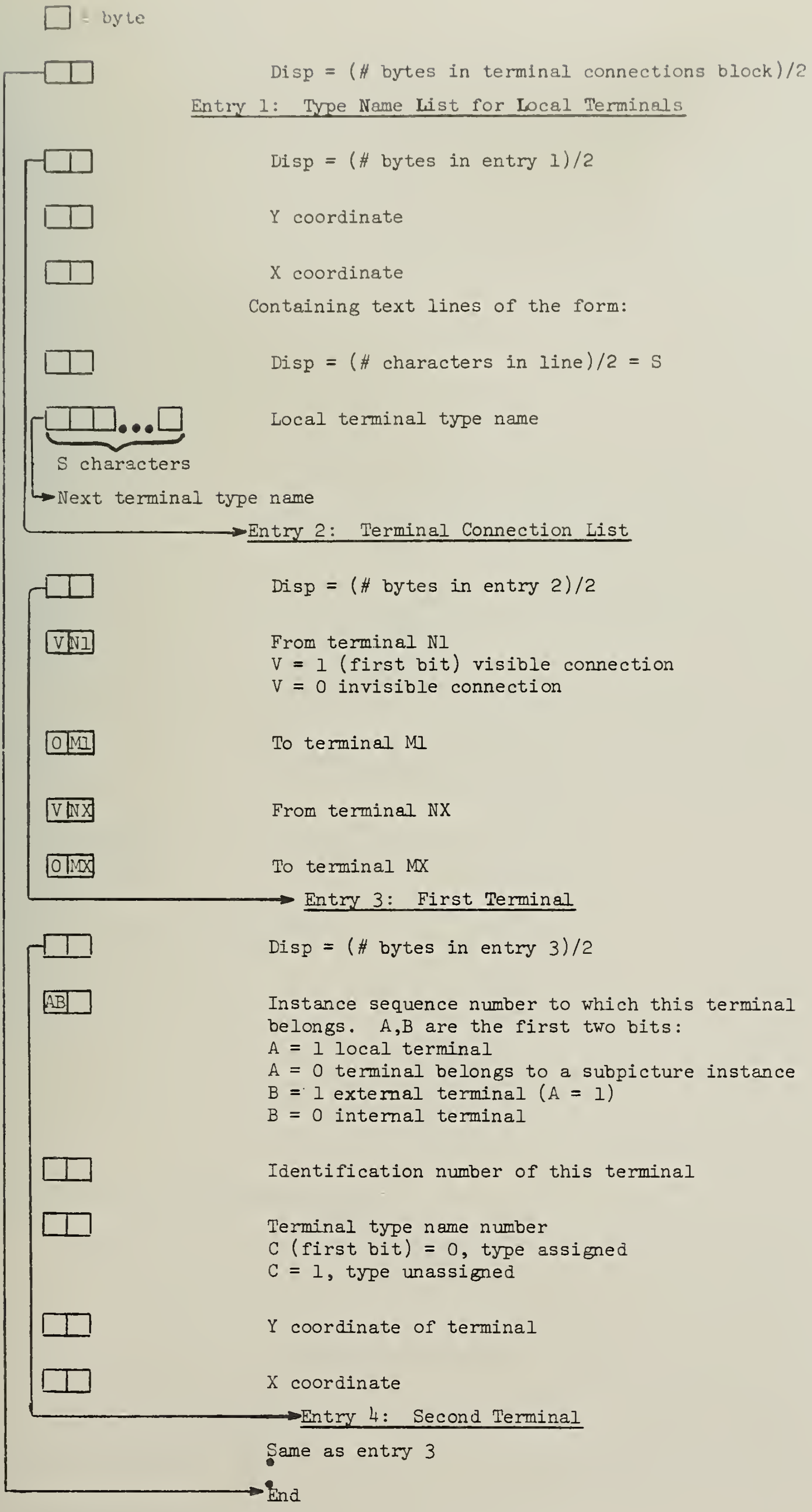
Disp = (# of bytes in node type name text)/2 = R

R characters

Next node type name

Next subpicture

Figure 5. Subpicture Block



Disp = (# bytes in terminal connections block)/2

Entry 1: Type Name List for Local Terminals

Disp = (# bytes in entry 1)/2

Y coordinate

X coordinate

Containing text lines of the form:

Disp = (# characters in line)/2 = S

Local terminal type name

Next terminal type name

Entry 2: Terminal Connection List

Disp = (# bytes in entry 2)/2

From terminal N1
V = 1 (first bit) visible connection
V = 0 invisible connection

To terminal M1

From terminal NX

To terminal MX

Entry 3: First Terminal

Disp = (# bytes in entry 3)/2

Instance sequence number to which this terminal belongs. A,B are the first two bits:
A = 1 local terminal
A = 0 terminal belongs to a subpicture instance
B = 1 external terminal (A = 1)
B = 0 internal terminal

Identification number of this terminal

Terminal type name number
C (first bit) = 0, type assigned
C = 1, type unassigned

Y coordinate of terminal

X coordinate

Entry 4: Second Terminal

Same as entry 3

End

Figure 6. Terminal Connection Block

3.4 Declaration Block

The declaration block has the form of a text block (Figure 7) with three entries. The first contains declarations of the E- and I-type variables associated with terminal types (see section 4.3). The second has the list of global variables and the third has the local variables (section 4.4).

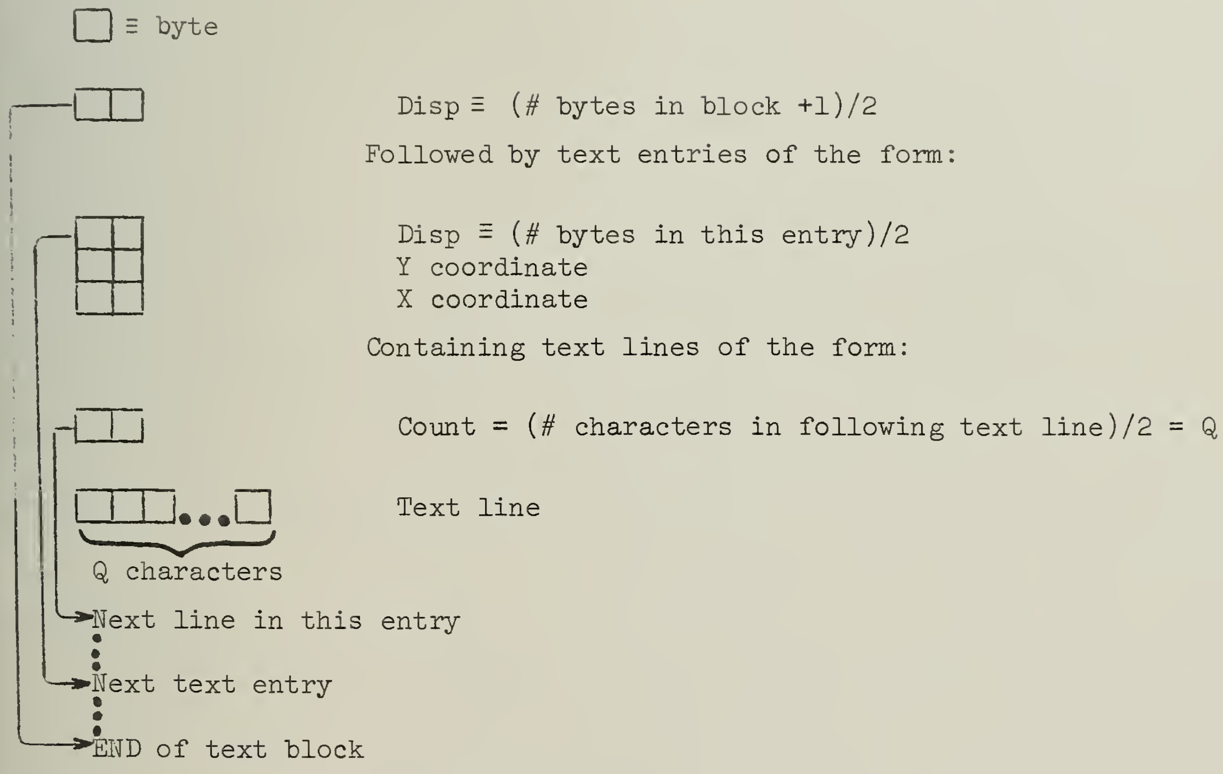

□ ≡ byte
Disp ≡ (# bytes in block +1)/2
Followed by text entries of the form:
Disp ≡ (# bytes in this entry)/2
Y coordinate
X coordinate
Containing text lines of the form:
Count = (# characters in following text line)/2 = Q
Text line
Q characters
Next line in this entry
Next text entry
END of text block

Figure 7. Text Block

3.5 Equation and Parameter Block

The last block is also a text block but with two entries (Figure 7). The parameters are listed in entry one and are used by ITEM to generate the list of parameters and default equations (section 4.5). Entry two has all the equations associated with the whole network. These are parsed by ITEM and their coded binary tree representation is placed in the output structure (sections 4.1, 4.2).

4. OUTPUT DATA STRUCTURES

Item Analysis is designed to translate certain data into a representation suitable for Global Analysis. These tables are placed on the filing system for use in communication between the user and the system. The items that are just translated and checked for syntactic errors are the user definition of terminal types, global variables, local variables and parameters. As shown in Figure 8, the lists of eight character (byte) names of globals, parameters, locals, double precision constants, left hand and EI type variables are referenced by counters and pointers at the topmost level of the output structure. Equations are handled both for the entire picture and for each element within that picture. The network is broken down into the pointers and tables shown in Figure 10. Each element becomes an entry in the Element Instance table. Finally, the whole structure is assembled and written onto ITEMLIB under the picture name.

4.1 Equations

All equations are parsed into a tree structure in a separate routine called PARSE. A top-down parse was used because of the relative ease of programming, debugging and modifying. Also, this method allowed us to isolate certain special case equations for special handling (section 4.2).

The top level consists of a string of pointers linking all the equations of a certain type together (default, network, parameter or element equations). The first word in each equation contains the length of the equation (i.e., displacement to the next equation) or '0'

□ ≡ byte

+0	*□□□□	length of GETMAIN = X'00002000'
+4	□□□□□□□□	picture name--6 characters and two EBCDIC blanks
+12	□□	length of remaining structure (in bytes)
+14	□□	Disp to the network (Figure 10) (NETP)
+16	□□	Disp to equations
+18	□□	length of equations
+20	□□	Disp to default equations
+22	□□	length of default equations
+24	□□	# of entries in terminal type list
+26	□□	Disp to terminal type list
+28	□□	# of global variables
+30	□□	Disp to list of globals
+32	□□	# of parameters
+34	□□	Disp to list of parameter names
+36	□□	# of local variables
+38	□□	Disp to local variable list
+40	□□	# of double precision constants
+42	□□	Disp to constant list
+44	□□	# of left-hand-side variables
#46	□□	Disp to list of left-hand variables
#48	□□	# of E- and I-type variables
#50	□□	Disp to EI list
#52	□□□□□□□□	Name of element
#60	□...□	All lists, equation trees and network definition

* All displacements are from +Ø

Figure 8. Output Data Structure

indicating this is the last equation in this chain. The first two bits of the second word are used to indicate the equation type:

00--element equation	resolved in Global Analysis
01--network equation	
10--default equation	resolved in Item Analysis
11--parameter equation	

The remainder of the first two bytes of the second word is the equation number in the indicated block. The last two bytes are used by Global Analysis for the displacement from the top of the interval tree to the entry associated with this equation's element. The rest of the words contain the equation in tree form. Each node of the tree consists of two or three halfwords.

Ø	L	R

Ø	L

Where Ø is the operation or operand and R, L are the right and left operands, respectively. As an operand, two halfwords are used; L indicates the number of the element in the indicated list. Ø can be one of the following:

X'01'--Global variable

X'02'--Parameter or left-hand variable

X'03'--Local variable

X'04'--E or I-type variable

X'06'--Double-precision constant

As an operation, R and L are displacements from the beginning of the equation to the operands. The legal codes for Ø are

X'07'--Equal sign

X'08'--Addition

X'09'--Subtraction

X'0A'--Multiplication

X'0B'--Division

X'0C'—Assignment

X'0D'--Exponentiation

X'10'--Differentiation

X'11'--Unary minus

Differentiation and unary minus are two while the others are three halfwords long. Special operators are

X'12'--System function

X'13'--User function

For the system function, the L halfword contains one of the following codes.

X'01'--Natural logarithm--LOG

X'02'--Exponentiation (natural base, e, raised to a power)--EXP

X'03'--Square root--SQRT

X'04'--Arctangent--ATAN

X'05'--Absolute value--ABS

X'06'--Cosine--COS

X'07'--Sine--SIN

L in the user function contains a count of the number of operands needed (say N), followed by N halfwords which contain displacements to the operands. All blanks in the equations are ignored. The first operation (root of the tree) is the equal sign which begins in the third word of the equation.

4.2 Parsing Equations

The parsing routine, PARSE, is set up as a subroutine called by Item Analysis. This was done since it is possible for equations to be present in several places in the input structure. When PARSE is called, register 1 points to the list of parameters depicted in Figure 9. PADDR points to the equation to be parsed.

Since PARSE works in a top-down manner, the error messages returned to the user are general ones. For example, "******ILLEGAL IDENT DETECTED" would refer to a specific equation but not to a specific identifier within that equation.

PARSE uses a register stack to allow it to make a guess and then easily backtrack if that guess was incorrect. Each type of statement has a separate routine that stores registers and updates the stack pointer on entry. Registers 1 and 10 are used to point at the current position in the equation. When an incorrect guess is made, the calling routine's registers are restored, thus moving the pointers back. When a successful guess is completed, all registers but 1 and 10 are restored. If an error is found, a message is printed and put on ERRLIB and a successful exit is made (not restoring registers 1 and 10). This was done to cause PARSE to skip over the bad areas in an equation and to catch other errors in the remainder of the equation if they exist.

Equations are parsed in the normal fashion until the special cases of LOG or SQRT are discovered. These functions have caused problems since they have points of singularity and complex results. When searching for an initial steady state, the "random" initial values used could cause these functions to cross these points of singularity

and abort the system. For each occurrence of LOG and SQRT, an extra variable and equation are generated.

User's equation	Output equations
A = SQRT(B) + C	A = KOTCHØØ1 + C
	KOTCHØØ1*KOTCHØØ1 = B
L = M*LOG(N)	L = M*KOTCHØØ2
	EXP(KOTCHØØ2) = N
X = LOG(SQRT(Y))/Z	X = KOTCHØØ3/Z
	EXP(KOTCHØØ3) = KOTCHØØ4
	KOTCHØØ4*KOTCHØØ4 = Y

Then it is possible to constrain these variables to be initially positive.

The Backus Normal Form description of the parse used is in the Appendix.

```
PARSPARM  DS   OF
          DS   X
PADDR     DS   AL3                ADDR OF WHAT I AM TO SCAN.
PEND      DS   A                  END OF WHAT TO SCAN.
VLOC      DS   A                  LOC OF LOCAL VARIABLES.
VGLB      DS   A                  GLOBAL.
VPAR      DS   A                  PARAMETERS.
VEI       DS   A                  E/I NAMES.
VLH       DS   A                  LH SIDE VARIABLES.
VCON      DS   A                  CONSTANTS.
FRES      DS   A                  WHERE TO GET FREE SPACE.
LASTEQN   DS   A                  ZERO WHEN STARTING NEW EQN TREES.
COUNT     DS   A                  EQN NUMBER IN BLOCK.
S8WORK    DS   A                  ADDR OF WORK AREA FOR MESS.  TO PDP8.
ERRLIST   DS   A                  ADDR OF ERRLIB DCB FOR PARSE ERRORS.
*                                 POSSIBLE FLAG BITS:
VLHBIT    EQU  X'80'              ONLY 1 VARIABLE ON LEFT SIDE OF =
HALTBIT   EQU  X'40'              ON IF PARSE HALTED.
ERRBIT    EQU  X'20'              ON IF ERRORS DETECTED BY PARSE.
MOREQN    EQU  X'08'              ON IF MORE EQNS GENERATED BY PARSE.
EXTRAS    EQU  X'04'              ON IF PARSING THOSE EXTRA EQNS.
PFLG      DS   X                  WHERE THE ABOVE FLAGS ARE TURNED ON.
TRANS     DS   AL3                ADDR OF EBCDIC TO ASCII TRANS.  TABLE.
FILE      DS   A                  ADDR USED BY XAPPEND IN PARSE
PTR       DS   A                  ADDR OF 1ST AVAIL POS FOR EXTRA EQNS
END       DS   A                  ADDR OF END OF ABOVE AREA
```

Figure 9. Parameters for PARSE

4.3 User Terminal Type Names

The simulation system allows the user to pass variables from terminal to terminal through the use of E- and I-type variables. E- and I-type variables are those that obey Kirchoff's voltage and current laws, respectively, and they refer to terminals by using the terminal number as a subscript (e.g., VOLT(1), VOLT(3), CURRENT(7)). To ensure that the correct variables are passed, only terminals of the same type may be joined. The user declares a terminal type with the following syntax.

```
    terminal type name :E( E-type variables ),
                         I( I-type variables )
```

For example, an electrical terminal might be declared as follows:

```
    ELECTRIC: E(VOLT, X), I(CURR, Y)
```

Then each ELECTRIC terminal would have two E-type (VOLT and X) and two I-type (CURR and Y) variables associated with it. Item Analysis takes the terminal type declarations and writes them onto NODLIB. There they can be accessed by Global Analysis to determine if the E- and I-type variables in the equations refer to the correct terminals. Item also uses the user assigned terminal type names in the Node-Connection table (section 4.7).

4.4 Global and Local Variables

Global variables are those that are associated with a whole element rather than a terminal. These variables are accessible by the network and have the same value for all elements at a particular level in a network. Local variables are those that are also associated with the whole element but are not accessible from the network. The user can

either enter globals and locals one to a line or more than one to a line separated by commas.

```
ALPHA
BETA                    or              ALPHA,BETA,GARBAGE
GARBAGE
```

Also note that any variables not assigned a type (global, local, special or parameter) will end up as local variables. Item Analysis prepares a list of the names of global and local variables and a count of each and places them in the output data structure (Figure 8).

4.5 Parameters

Parameters are those variables that can have a different value for each instance (occurrence) of an element (e.g., resistance, weight). Since a value must be specified for each occurrence of an element, the user is allowed to assign a default value to a parameter. Thus, if a resistor with a default resistance of 100 is used in a network and no new resistance is specified, its resistance will remain 100. Parameters are entered in the same manner as globals and locals but may have default values associated with them. The following are legal parameter declarations.

```
REST,TEMP=30.7,VAR
ORANGE
MONK=763+GVAR
```

A list and count of parameters is made by Item Analysis and it must also check for default equations. When a default equation is found, it must have only one variable on the left side of the equal sign. It is parsed

by Item (section 4.1, 4.2) and its binary tree representation placed in the default equations section of that element (Figure 8).

4.6 Left-hand-side Variables

Left-hand-side variables occur only in the default equations. Any variable occurring on the "left-hand-side" of a default equation will be put in this list. This causes the variable to be assigned the value occurring on the right-hand-side during the simulation. Otherwise, the variable's value would be allowed to change when the system tried to "balance" the values on both sides of the equal sign. Thus, the equal sign is changed to an assignment operator.

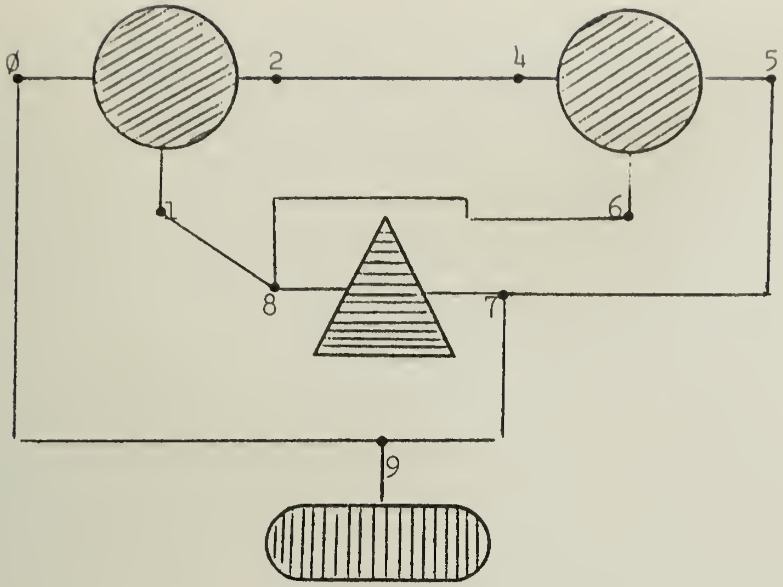
4.7 Network Output

A set of pointers and tables that completely describe the structure of the network is pointed to by NETP (Figure 8). If the picture analyzed is a basic element, NETP = 0 and these pointers do not exist in the output structure. Item Analysis receives as input a list of connections (e.g., Ø-9, 1-8, 9-7, 5-7, 8-6, 2-4) and the correspondence between terminals and elements. ITEM resolves these into a Node/Connection table.

+Ø**	# of nodes in network (NNODES)
+2	# of connections in network (NCONNS) and Node/Connection table
+4	Disp to Node/Connection table
+6	# of element instances
+8	Disp to Element Instance table
+10	length of network equation block (in bytes)
+12	disp to network equations
+14	number of external terminals
+16	position numbers in the Node/Connection table of the external terminals
+18	
⋮	

** All displacements in this section are from +Ø.

Figure 10. Network Output



Type	Terminal
CONTROL	Ø, 5, 7, 9, 2, 4
WATER	1, 6, 8

Node/Connection Table

Position # (not present in table)	Type	Next	User's
Ø	CONTROL	3	5
1	WATER	6	1
2	CONTROL	8	4
3	Ø	4	Ø
4	Ø	5	7
5	Ø	Ø	9
6	1	7	6
7	1	Ø	8
8	2	Ø	2

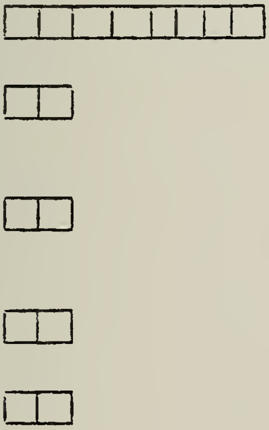
Figure 11. Example

A node in this context is composed of a group of terminals that are interconnected. For instance, in Figure 11, there is a node composed of terminals Ø, 9, 7, 5 and one of 1, 8, 6 and a third of 2, 4. So there are three entries in the node part of the table and NNODES = 3. Linked to each entry in the node portion are the rest of the terminals that are interconnected. In the example, NEXT contains the position number in the table of the next terminal that is on this node. If NEXT = Ø, it is the last terminal in the chain. USERS are the original terminal numbers assigned when the user originally drew the picture. The first field of the terminals in the node part of the table is an eight character name of the terminal type assigned to this node. Global Analysis checks to see that all terminals on a node are of the same type. The first field in the connection part of the table is a position number referring back up to the node to which this terminal is connected. Thus, in the example, position 4 in the table (originally terminal #7) is connected to node Ø. The remaining two fields in the connection part are the same as in the node part. For the example, NCONNS = 6 since there are six entries in the connection part of the table. The position numbers in the Node/Connection table are used later when a list of the terminals on each element is constructed. This was done to facilitate a search by Global Analysis through the network.

Item Analysis also constructs an Element Instance table (Figure 12). This is a list of all the elements that were used to construct the network. Each entry in the table consists of an eight character element name and four halfwords. The halfwords indicate the length of the assigned parameter equations and their position and the

number of terminals on the element and a displacement to a list of them, respectively. Each element was given a separate parameter block because of the possibility of two elements of the same type being used; for instance, if two resistors were used but in the first instance the parameter resistance was assigned a value of 100 while in the second instance it had a value of 200. In this manner, the two parameter blocks can be distinguished.

The next two halfwords in the network output section (Figure 10) are the length of and displacement to the network equation tree. These refer to the equations that were written that reference the entire network. Finally, there is a list of the position numbers in the Node/Connection table of all external terminals.



Element name

Length of the parameter equations
associated with this element

Disp from network base (+0 Figure 10)
to parameter equations

# of terminals on element

Disp from network base to list of
position numbers in Node/Connection
table corresponding to the terminals
on element

Figure 12. Element Instance Table Entry

LIST OF REFERENCES

[1] Gear, C. W. An Interactive Graphic Modeling System, Department of Computer Science Report No. 318, University of Illinois, Urbana, Illinois 61801 (April 1969).

[2] Gear, C. W., et. al. The Simulation and Modeling System--A Snapshot View, Department of Computer Science File No. 824, University of Illinois, Urbana, Illinois 61801 (February 1970).

[3] Michel, M. J. GRASS: System Software Description, Department of Computer Science Report No. 468, University of Illinois, Urbana, Illinois 61801 (August 1971).

[4] Michel, M. J. and Haskin, R. GRASS: Extended Remote Facilities Guide, Department of Computer Science File No. 867, University of Illinois, Urbana, Illinois 61801 (August 1971).

[5] Michel, M. J. and Koch, J. A. GRASS: Terminal User's Guide, Department of Computer Science Report No. 467, University of Illinois, Urbana, Illinois 61801 (August 1971).

APPENDIX

DEFINITION OF PARSE

```
<ASSIGN> ::= <IDENT> = <EXP>

<EXP> ::= <TERM> <EXP FRAG>

<EXP FRAG> ::= <PTERM> <EXP FRAG>|<MTERM> <EXP FRAG>|<NULL>

<NULL> ::=

<TERM> ::= <FACTOR> <FACTOR FRAG>

<FACTOR FRAG> ::= <MULFACTOR> <FACTOR FRAG>|<DIVFACTOR> <FACTOR FRAG>|<NULL>

<PTERM> ::= +<TERM>

<MTERM> ::= -<TERM>

<FACTOR> ::= <DIFFER> <AFACT>|<DIFFER>

<PRNEXP> ::= (<EXP>)

<AFACT> ::= **<FACTOR>

<DIVFACTOR> ::= /<FACTOR>

<MULFACTOR> ::= *<FACTOR>

<DIFFER> ::= <POWER>''''.....'|
              .
              .
              .
             <POWER>'''|

             <POWER>'' |

             <POWER>'  |

             <POWER>

<POWER> ::= <ICON>|+<ICON>|-<ICON>|<PRNEXP>|-<PRNEXP>|+<PRNEXP>

<ICON> ::= <IDENT>|<CONSTANTS>

<SUBSCRIPT> ::= (<NUMBER><NUMBER>...<NUMBER>)

<NUMBER> ::= Ø|1|2|3|4|5|6|7|8|9

<LETTER> ::= A|B|C|D|...|W|X|Y|Z

<ALPHANUM> ::= <LETTER>|<NUMBER>
```

```
<CONSTANTS> ::= <NUMBER>...<NUMBER> E <NUMBER><NUMBER>|
                <NUMBER>...<NUMBER> D <NUMBER><NUMBER>|
                <NUMBER>...<NUMBER>
<IDENT> ::= MAX (<EXP>, <EXP>,..., <EXP>)|MIN (<EXP>,..., <EXP>)|
            SIN (<EXP>)|COS (<EXP>)|ABS (<EXP>)
            ATAN (<EXP>)|SQRT (<EXP>)*|EXP (<EXP>)|
            LOG (<EXP>)*|<LETTER> FOLLOWED BY FROM 0 TO 7 <ALPHANUM>'s
           <SUBSCRIPT>|<LETTER> FOLLOWED BY FROM 0 TO 7 <ALPHANUM>'s
```

* 'LOG' and 'SQRT' are special cases (see section 4.2)

n AEC-427
(6/68)
ECM 3201

U.S. ATOMIC ENERGY COMMISSION
UNIVERSITY–TYPE CONTRACTOR'S RECOMMENDATION FOR
DISPOSITION OF SCIENTIF:C AND TECHNICAL DOCUMENT

(See Instructions on Reverse Side)

AEC REPORT NO. COO-1469-0201	2. TITLE ITEM ANALYSIS

TYPE OF DOCUMENT (Check one):

☒ a. Scientific and technical report
□ b. Conference paper not to be published in a journal:
Title of conference ______
Date of conference ______
Exact location of conference ______
Sponsoring organization ______
□ c. Other (Specify) ______

RECOMMENDED ANNOUNCEMENT AND DISTRIBUTION (Check one):

☒ a. AEC's normal announcement and distribution procedures may be followed.
□ b. Make available only within AEC and to AEC contractors and other U.S. Government agencies and their contractors.
□ c. Make no announcement or distrubution.

REASON FOR RECOMMENDED RESTRICTIONS:

SUBMITTED BY: NAME AND POSITION (Please print or type)

C. William Gear
Principal Investigator

Organization

Department of Computer Science
University of Illinois
Urbana, Illinois 61801

Signature Charles W Gear.	Date March 1972

FOR AEC USE ONLY

AEC CONTRACT ADMINISTRATOR'S COMMENTS, IF ANY, ON ABOVE ANNOUNCEMENT AND DISTRIBUTION RECOMMENDATION:

PATENT CLEARANCE:

□ a. AEC patent clearance has been granted by responsible AEC patent group.
□ b. Report has been sent to responsible AEC patent group for clearance.
□ c. Patent clearance not required.

.IOGRAPHIC DATA ET	1. Report No. UIUCDCS-R-72-507	2.	3. Recipient's Accession No.

tle and Subtitle

ITEM ANALYSIS

5. Report Date
March 1972

6.

thor(s) John Allen Koch

8. Performing Organization Rept. No.

rforming Organization Name and Address

Department of Computer Science
University of Illinois
Urbana, Illinois 61801

10. Project/Task/Work Unit No.
US AEC AT(11-1)1469

11. Contract/Grant No.
US AEC AT(11-1)1469

onsoring Organization Name and Address

Argonne National Laboratory
9600 S. Cass Avenue
Argonne, Illinois

13. Type of Report & Period Covered
Master's Thesis

14.

ipplementary Notes

Master's Thesis Research Report

ostracts

This thesis describes Item Analysis, a program that analyzes a user's description of a network and converts it into a structure that is suitable for analysis. This program is contained in the graphics-oriented simulation and modeling system developed at the University of Illinois. This report also contains documentation of the program which may not be of interest to the reader who only needs a broad description of the function of Item Analysis. Such readers are encouraged to cover only the first section under each major heading.

y Words and Document Analysis. 17a. Descriptors

Item Analysis
Local Variables
Global Variables
Global Analysis

lentifiers Open-Ended Terms

OSATI Field/Group

Ailability Statement Release unlimited	19. Security Class (This Report) UNCLASSIFIED	21. No. of Pages 45
	20. Security Class (This Page UNCLASSIFIED	22. Price